

Configuration Management Plan

for the

Comprehensive Large Array-data Stewardship System

(Version 1)

October 1, 2002

Revisions

Version	Description of Version	Date Completed
Draft 0.1	Initial draft	7/05/02
Draft 1.0	Updated with QMO Review Comments – ready for CPMT review	7/16/02
1.0	Incorporated comments from CPMT – approved by CPMT	10/01/02
1.0	Cleanup non-printing characters	10/07/02

Review & Approval**Project Plan Review History**

Reviewer	Version Reviewed	Signature	Date
Constantino Cremidis/CSC			
Alexander Kidd/OSDPD			
Geof Goodrum/NCDC			
Carlos Martinez/TMC			
Ted Habermann/NGDC			
Eric Kihn/NGDC			
David Vercelli/NESDIS			
Robert Mairs/NESDIS			
Anita Konzak/CSC/ QMO			

Table of Contents

1	INTRODUCTION.....	1
1.1	Purpose.....	1
1.2	Scope.....	1
1.3	Applicable Documents	1
1.4	Acronyms	1
1.5	Definitions.....	2
2	ORGANIZATION	3
2.1	Distributed Development Team.....	3
2.2	Roles and Responsibilities	4
2.2.1	Responsibilities of the CCB.....	4
2.2.2	Responsibilities of the CCB Chair.....	5
2.2.3	Responsibilities of the Configuration Management Officer (CMO)	5
2.2.4	Responsibilities of the CLASS Lead Integrator.....	5
3	CONFIGURATION DEFINITION	5
4	CONFIGURATION CONTROL	6
4.1	Identification of Proposed Changes	6
4.2	Documentation of Proposed Changes	7
4.3	Evaluation of Proposed Changes.....	7
4.3.1	Processing Emergency Changes	7
4.3.2	Processing Urgent and Routine Changes.....	8
4.4	Integration of Approved Changes	8
5	CONFIGURATION STATUS ACCOUNTING	8
5.1	Status Account Data	8
5.2	CLASS Configuration Management Database	11
6	CONFIGURATION AUDITS.....	11
6.1	CM Process Audits.....	11
6.2	Functional Audits.....	11
6.3	Physical Audits.....	11
	APPENDIX A: DETAILED SOFTWARE BUILD AND PROMOTION PROCESS.....	12

Comprehensive Large Array-data Stewardship System Configuration Management Plan

1 Introduction

1.1 Purpose

The purpose of this Configuration Management (CM) Plan is to identify and define the organization, activities, overall tasks, and objectives of Configuration Management for the Comprehensive Large Array-data Stewardship System (CLASS). This initial version addresses the following major CM activities: change request management, software build and promotion processes, and CM audits. Additional CM-related processes and procedures may be defined as the CLASS project matures. As they are defined and approved, they will be documented in this or related documents as CLASS project standards.

1.2 Scope

This plan is applicable to all systems managed by the CLASS Project Management Team (CPMT), as well as the associated hardware and software used to support those systems.

1.3 Applicable Documents

CLASS System Requirements (September 24, 2002)

CLASS Allocated Requirements (TBW)

CLASS System Architecture (TBW)

CLASS Software Development Guide (draft, July 22, 2002)

CLASS Quality Management Plan (draft, August 30, 2002)

1.4 Acronyms

CCR	Configuration Change Request
CCB	Configuration Control Board
CI	Configuration Item
CIO	Chief Information Office
CLASS	Comprehensive Large Array-data Stewardship System
CM	Configuration Management
CMDB	Configuration Management Database
CMO	Configuration Management Office
CPMT	CLASS Project Management Team
CVS	Concurrent Versions System
IPD	Information Processing Division
NCDC	National Climatic Data Center
NESDIS	National Environmental Satellite, Data, and Information Service
NGDC	National Geophysical Data Center

NOAA	National Oceanic and Atmospheric Administration
OSDPD	Office of Satellite Data Processing Division
PAL	Project Area Lead
PR	Problem Report
SAA	Satellite Active Archive
SET	System Engineering Team (CLASS)
TAL	Technical Area Lead
TBD	To Be Determined
TBW	To Be Written

1.5 Definitions

Baseline	A formal, approved document or product that serves as a departure point for future releases. The CLASS baseline includes a documentation baseline and the system baseline (software, hardware, etc.)
CLASS Configuration Control Board	The board that defines the disposition of Configuration Change Requests. The board is composed of CPMT members and their designees, and SET members.
CLASS Project Management Team	The managing authority for the CLASS project. This team is defined in Section 2.
CLASS Systems Engineering Team	The technical advisory committee for CLASS. This team is defined in Section 2.
Configuration Change Request	A request for change to a baseline document or system.
Configuration Item	An aggregation of hardware, software, or both that is designated for configuration management and treated as a single entity in the configuration management process.
Level-I CCR	A request for a change that requires changes to the CLASS System Requirements document or the CLASS System Architecture document.
Level-II CCR	A request for change that does not require updates to the CLASS Requirements document or the CLASS System Architecture, but does require a change to the allocated requirements or interfaces.

Level-III CCR	A request for change that does not require updates to the CLASS Requirements document, the CLASS System Architecture, the allocated requirements, or interfaces.
Originator	The person who submits a Configuration Change Request.
Problem Report	A request for a change submitted to the Remedy Configuration Management tool, documenting a problem identified during system integration and test.

2 Organization

2.1 *Distributed Development Team*

The CLASS project is being conducted in support of the mission of the National Environmental Satellite, Data, and Information Service (NESDIS) to acquire, archive, and disseminate environmental data. A distributed team that includes NOAA personnel at OSDPD, NCDC, and NGDC; and contractors in Suitland, MD, and Fairmont, WV, is developing the system. The development of CLASS is expected to be a long-term, evolutionary process, as current and new campaigns are incorporated into the CLASS architecture. Development systems are located at OSDPD in Maryland, NCDC in North Carolina, NGDC in Colorado, and the NCDC contractor site in Fairmont, WV, with the integration system located at the OSDPD facility in Maryland. The system will operate at both the OSDPD facility in Suitland, MD, and the NCDC facility in Asheville, NC.

Due to the distributed and evolutionary characteristics of the project, the Contractor Technical Area Lead (TAL) and the Government Program Area Lead (PAL) from each participating organization serve on the CLASS Project Management Team (CPMT), led by the NESDIS Chief Information Officer (CIO). The CPMT is responsible for overall direction and coordination for CLASS. Similarly, each development team is represented on the CLASS Systems Engineering Team (SET), which is led by the CLASS Lead Integrator. The SET oversees the technical direction of the development to ensure consistency and compatibility among the various components.

In such a distributed development environment, it is particularly important that changes to the baseline system be strictly controlled. The Configuration Management Office (CMO) manages all proposed changes to the CLASS baseline (requirements, software, hardware, etc.), with the review and approval of the CLASS Configuration Control Board (CCB). The CLASS CCB

includes the members of the CPMT and the SET, the CMO, and additional members as designated by a TAL or PAL. This board reviews for approval all CLASS configuration change requests (CCRs). The Chair of the CCB is the OSDPD PAL, who has decision-making authority on the disposition of CCRs, with input from the other CCB members.

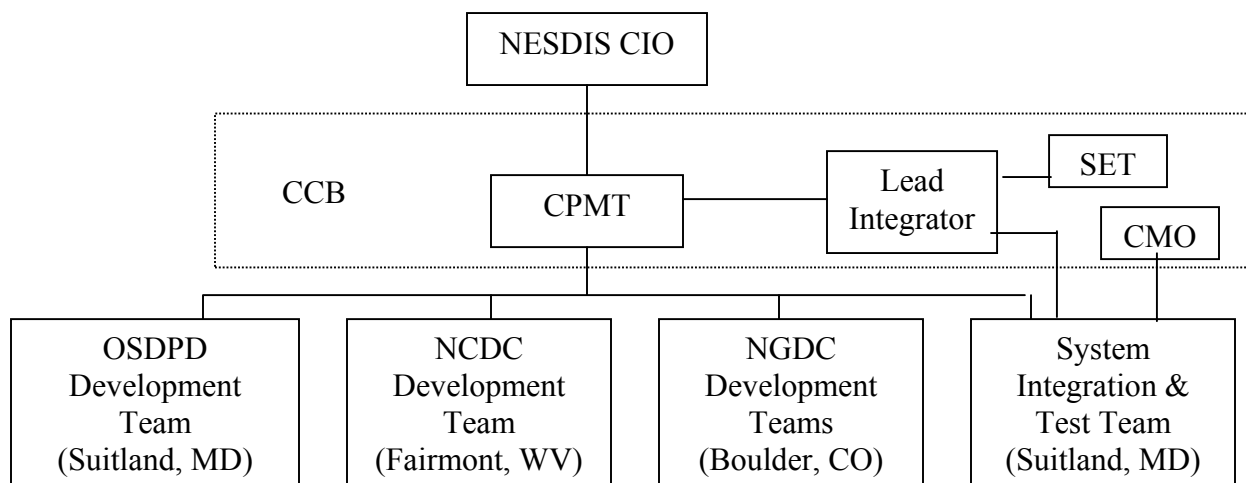


Figure 1 - CLASS Organization

When a CCR is received, the CLASS Lead Integrator (with input from the SET as needed) evaluates it, assigns it to the appropriate Level (I, II, or III) and priority, and provides an impact assessment for implementation. The CCB may adjust the level or priority if they believe that it has been incorrectly assigned. The CCB can approve or reject a CCR, escalate it to a higher authority, or assign it for further analysis. The CMO tracks all CCRs to ensure that they are acted on in a timely fashion.

2.2 Roles and Responsibilities

The major roles in configuration management activities are defined below.

2.2.1 Responsibilities of the CCB

- < Review all CCRs and provide the necessary input to determine the disposition (approve, reject, or escalate). If additional information is needed, assign the CCR to the appropriate group(s) for further analysis.
- < Assign approved CCRs to an implementation date, and function or subsystem.
- < Ensure that action is taken on change requests in a timely fashion.

2.2.2 Responsibilities of the CCB Chair

- < Direct the CCB meetings.
- < Assign/approve disposition of each CCR and implementation assignment for approved CCRs.
- < Manage escalation process when a CCR is referred for higher level approval or input.
- < Ensure that action is taken on change requests in a timely fashion.

2.2.3 Responsibilities of the Configuration Management Officer (CMO)

- < Maintain the CLASS CM Plan.
- < Identify Configuration Items (CI) and document their characteristics.
- < Control and facilitate changes to the characteristics of a CI.
- < Perform audits to verify compliance with CLASS CM Plan
- < Manage the configuration management database.
- < Report to the CCB approval status of all proposed changes and implementation status of all approved changes.
- < Work with the CCB Chair and members to schedule regular CCB meetings, and prepare the agenda for each meeting.
- < Perform system builds and promotions.

2.2.4 Responsibilities of the CLASS Lead Integrator

- < Monitor development for technical integrity and consistency, and compliance with the system requirements and architecture.
- < Review change requests as they are received; classify them as Level I, II, or III; review the assigned priority; and determine level of effort for implementation.
- < Work with the CMO to ensure that action is taken on change requests in a timely fashion.
- < Support the CCB in review and disposition of CCRs.
- < Lead system integration and test for each release.
- < Review Problem Reports to resolve questions and priority.

3 Configuration Definition

Configuration-managed items are those items directly involved with the system and therefore part of the baseline. All aspects of configuration management apply to these items. These comprise:

- The deliverable system at all levels, from configuration items down to configuration units.
- Related documents—Documents that define configuration items and components, such as negotiated agreements, specifications, charts, drawings, code, and change orders.
- Development hardware and software.

The CMO defines the CIs that constitute the deliverable system. At the top level, CLASS is treated as a CI – changes are managed, tracked, and reported at the CLASS level. To provide greater visibility to the stability of the system and its components, the CMO defines lower level CIs such as subsystems, functional areas, hardware components, and, at the lowest level, individual code units. This allows version control and reporting at various levels within CLASS. The Configuration Management Database (CMDB), described in Section 5.2, maintains the key characteristics of each CI, as defined by the CMO (e.g., software unit version numbers, hardware model numbers, system release numbers).

CIs for the Satellite Active Archive (SAA) are currently defined in the CMDB, and these form the basis for the CLASS configuration. The CMO will review and update the CI definitions when the CLASS system architecture is completed and approved, and as additional CLASS components are designed.

The baseline configuration for CLASS and its subsystems is defined by the CLASS System Requirements, the CLASS System Architecture, the CLASS Allocated Requirements, and the Configuration Management Database. They contain performance criteria and design specifications for CLASS and its subsystems. This documentation baseline is the basis for future enhancements and will mature as CLASS continues to improve its systems. Once approved by the CPMT, these documents, along with the system they describe, constitute the approved CLASS baseline, and are subject to the processes defined in this CM Plan.

Some documentation on a project is working documentation that is not part of the deliverable system, and is therefore not included in baselines. This documentation, however, also needs to be managed and controlled to ensure that changes are made in an organized manner and that the current version is always known and available. Examples of such items are project plans (such as the configuration management plan, quality plan) and internal project procedures (such as inspection procedures). This documentation is not governed by the procedures defined in this CM Plan, however, changes to this documentation must be approved by the CPMT.

4 Configuration Control

Configuration Control is the systematic proposal, justification, evaluation, coordination, approval or disapproval of proposed changes. Configuration control also includes the implementation of all approved changes to the baseline. The Configuration Control process includes: Identification of proposed changes, documentation of proposed changes, evaluation and disposition of proposed changes, and integration of approved changes.

4.1 Identification of Proposed Changes

Changes are permanent alterations to the established baseline (system or documentation). Each change is designated as either a Level I, Level II, or Level III change. Changes requiring changes in the CLASS System Requirements document or in the CLASS System Architecture document

are classified as Level I changes. Changes that do not fall into the Level I category, but require changes to the CLASS Allocated Requirements or CLASS interfaces are classified as Level II. All other changes are classified as Level III changes.

A change is proposed when a new requirement is received, an improvement is desired, or a problem requires solution. Changes may be requested by Government or contractor personnel.

Problems identified during system integration and test, and resolved prior to delivery of the system to operations, are handled by the development and test teams as Problem Reports, under the direction of the Lead Integrator (with CPMT oversight if necessary). Problems identified during system integration and test that are not resolved prior to delivery become requests for changes to an established system baseline (i.e., the production system), and are converted to CCRs when the release is delivered. These CCRs must be approved by the CPMT when the release is approved for promotion to operations. These CCRs are scheduled for future releases and tracked in the same way as other CCRs.

4.2 Documentation of Proposed Changes

The Remedy Configuration Management tool is used to submit proposed changes. The instructions for submitting proposed changes are available via the online help pages provided with the tool.

4.3 Evaluation of Proposed Changes

Each CCR submitted via the Remedy tool is directed to the CLASS Lead Integrator. The Lead Integrator reviews the CCR, classifies it as a Level I, Level II, or Level III change, reviews or assigns a priority, and provides an impact assessment (a rough estimate of the level of effort required for implementation, and impact to other current and planned activities). The Lead Integrator then works with the CMO to schedule the CCR for CCB review.

One of three change priorities is assigned to every change request: Emergency, Urgent or Routine. All changes regardless of their priority follow the same approval process. The main difference is the time allowed for review. The priority of a CCR is assigned either by the originator or by the Lead Integrator. The Lead Integrator has the authority to alter the priority of any CCR.

Upon receipt of a new change request, the CCB evaluates the change, contacts the originator of the change if needed, approves or denies the request for a change, and schedules the change for implementation.

4.3.1 Processing Emergency Changes

A CCR is assigned an Emergency priority when the originator or the Lead Integrator determines that a delay in the implementation would impact operations or create software failures. Emergency CCRs are assigned to the Integration Facility TAL who will try to obtain approval

out of board from the CCB within two hours. If no resolution is achieved, the Integration Facility TAL will approve or deny the request.

4.3.2 Processing Urgent and Routine Changes

Urgent and Routine changes follow the same flow process, however urgent changes will generally be assigned to the next scheduled software release or may cause a software release to be scheduled. It is the responsibility of the Lead Integrator to obtain timely approval of urgent CCRs to be included in the next scheduled software release.

4.4 Integration of Approved Changes

The requirements manager makes changes to the baseline documentation for Level I and Level II CCRs immediately after CCB approval. Once the changes are made and the necessary traceability established in DOORS, the Lead Integrator verifies that the change was correctly documented and the CMO closes the CCR. Implementation is then tracked along with all other requirements.

The CCB assigns each approved Level III CCR to an implementation date and a functional group or subsystem when the CCR is approved. The assignment is reviewed by the CPMT during release planning to verify that the scope of the release is acceptable and consistent with current priorities. The Lead Integrator is responsible for ensuring that all scheduled changes are implemented, and for verifying the correct implementation of the changes. All verified CCRs are assigned to the CLASS CMO, who is responsible for collecting all change requests assigned to a software release and incorporating them into the operational system. Appendix A provides complete details of the process for this task. After the CMO has promoted the changes to the operational environment and verified that the changes are functioning correctly in that environment, the CMO closes the CCR.

Figure 2 shows the high-level process flow for managing change requests.

5 Configuration Status Accounting

Configuration Status Accounting includes the collecting, processing, maintaining and publishing data necessary to effectively manage the configuration.

5.1 Status Account Data

The CMO collects data necessary to produce reports useful to the CPMT, CCB, and Lead Integrator. Typical data includes:

- Identifying information pertaining to each Change Request received and its status in the Remedy Configuration Management tool.
- Identifying information pertaining to each controlled configuration item, i.e. current revision, revision history, associated subsystem, etc.

The CMO distributes bi-weekly reports with the status of each CCR in the system. The reports include summary sections detailing: new change requests, newly approved change requests, recently closed change requests and a list of change requests to be included in the next software release.

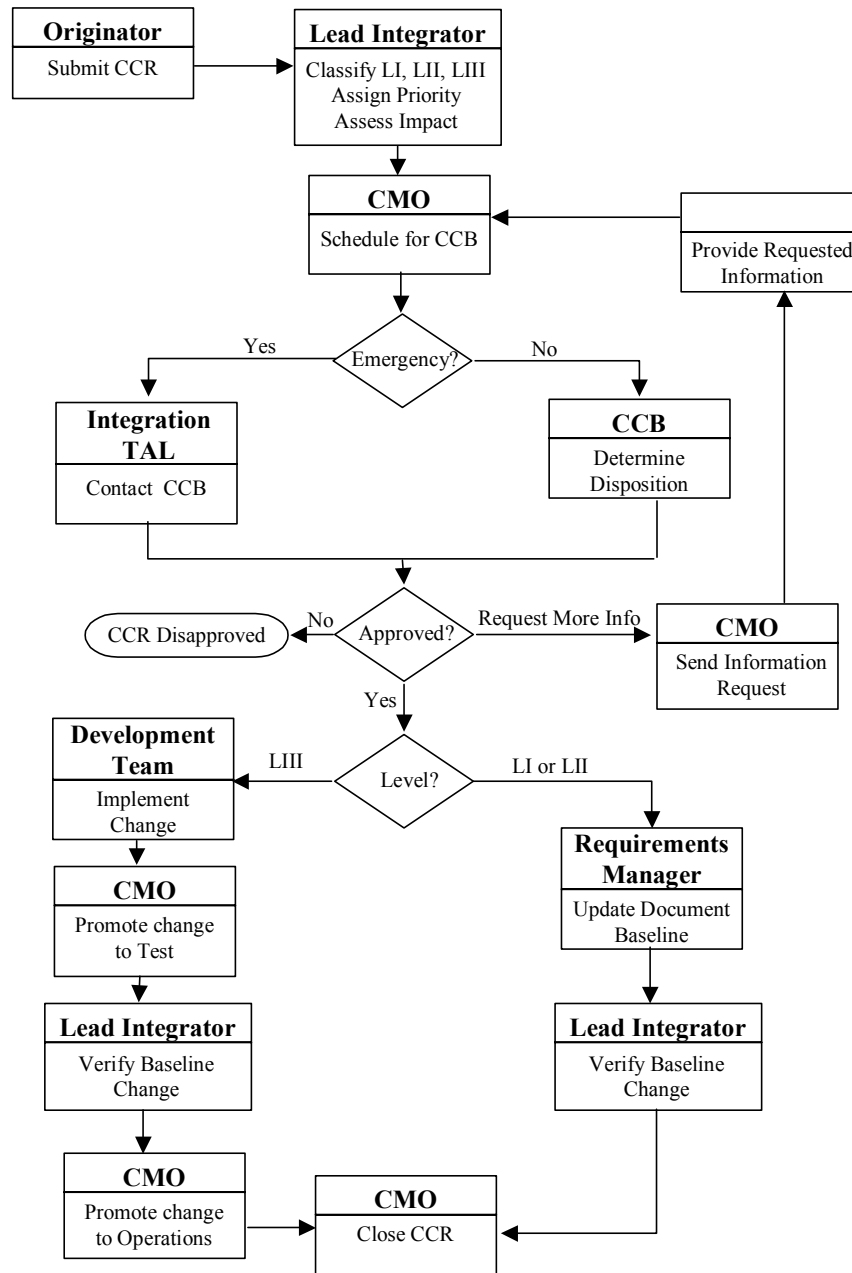


Figure 2 - CCR Process Flow

5.2 CLASS Configuration Management Database

The CLASS configuration management database contains detailed information about every CLASS configuration item; it is used for status reporting as well as for moving new CLASS software releases to operations. The CMO is responsible for maintaining, supporting and updating the CLASS configuration management database.

6 Configuration Audits

Configuration audits consist of reviews where the CM process or a product configuration is compared to requirements to determine if those requirements are being met.

6.1 CM Process Audits

Process audits confirm that the CM process is being followed. These audits focus on the processes being used rather than on the products being produced. They examine the manner in which the CM activities are performed against the documented procedures. The quality assurance office conducts these audits on a regular basis during the course of the project to allow for problem identification and corrective action. The CLASS Quality Assurance Plan provides further information regarding the CM process audit.

6.2 Functional Audits

Functional audits confirm that a product performs as required. They typically involve an examination of the test results against Test Plans and Materials to verify that the tests were executed as planned, and an examination of test results against the product's requirements to verify that the demonstrated functionality is consistent with the required functionality. The CMO conducts this audit before a release is approved for promotion to operations.

6.3 Physical Audits

Physical audits confirm that the product is consistent with its documentation. It examines the as-built version of a product (or CI) against the technical documentation that defines it to verify that the product and its documentation are consistent, all authorized changes have been made, and no unauthorized changes have been made. The CLASS CMO, along with the Lead Integrator, conducts this audit before a release is approved for promotion to operations.

Appendix A: Detailed Software Build and Promotion Process

1.1 Overview

In CLASS, configuration management is closely bound to the concept of a Configuration Change Request (CCR). No code is moved or modified without a CCR being opened to document and control its changes and movement.

The Remedy Change Management Tool, is used to create, update, and track the status of CLASS CCRs. The Concurrent Versions System (CVS) is used in the CLASS system to control access to source files and maintain configuration management information for each file. A set of in-house Perl scripts is used to promote software changes from one environment to another.

1.2 CLASS Environments

CVS is used to maintain the configuration management information for each file. This section gives a brief description of how CVS is set up and used for CLASS, and the environments used to transition changes from development to operations.

Software Repository

The CVS repository stores a complete copy of all the files and directories that are under version control. To access the files, the developer or CMO uses CVS commands to create a copy of the files into a working directory and then works on that copy. When the developer completes a set of changes, the developer *commits* them back into the repository. The repository then contains the changes made, as well as recording exactly what was changed, when it was changed, and other such information. Note that the repository is not a subdirectory of the working directory, or vice versa; they are in separate locations.

Development (Dev)

A separate development platform is in place at each development location. Programmers perform all software development in this environment. They can check out source files from CVS, modify source files, and commit changes. Programmers can create and test runtime files in the Development environment. Each night, a system build is run from the development source library to create the latest version of the development system on each development platform.

Integration (Int)

When a release, or a component of a release, is approved for turnover to the Integration and Test team, the CLASS CMO is responsible for marking the files for promotion to Integration. The CMO checks out delivered versions of source code to the Int environment, and runs *make* to create runtime files in this environment.

Beta Test (Beta)

This environment includes runtime files (no source code) under directories corresponding to each operational processor. Each of these directories contains copies of all the runtime files that are hosted on one operational processor. The directory on the test machine serves as a staging and testing area for runtime files to be promoted to the corresponding operational machine.

The CLASS CMO is responsible for promoting runtime files from the Int environment to the appropriate Beta directory. To set up an environment that replicates the environment on a particular operational processor, the test manager logs in, sets his environment variable HOME to the Beta directory, and executes the .profile script in that directory.

Operations (Oper)

This environment includes CLASS runtime files distributed among the operational processors. The CLASS CMO is responsible for promoting runtime files from the Beta Test environment on the test machines to the corresponding directory on the operations processor.

A problem may be discovered in any environment, but software changes always start in the Development environment and are promoted through the other environments. Some files, such as those that define environment variables or configuration parameters, may have to be customized in each environment. This is done with special CM scripts.

1.3 Promotion Process

The promotion process is the sequence of steps followed from the creation of a Configuration Change Request (CCR) until the final disposition of this proposed change. All software changes follow the same process with the exception of emergency CCRs. Beta testing for emergency CCRs is streamlined, and the emergency CCR then goes directly from Beta Test to Operation; there is no wait for a scheduled release.

Figure 3 shows the process flow for the approved change request as well as the software promotion flow, as described in the following paragraphs. See Figure 2 and the discussion in Section 4 for the CCR review and approval process.

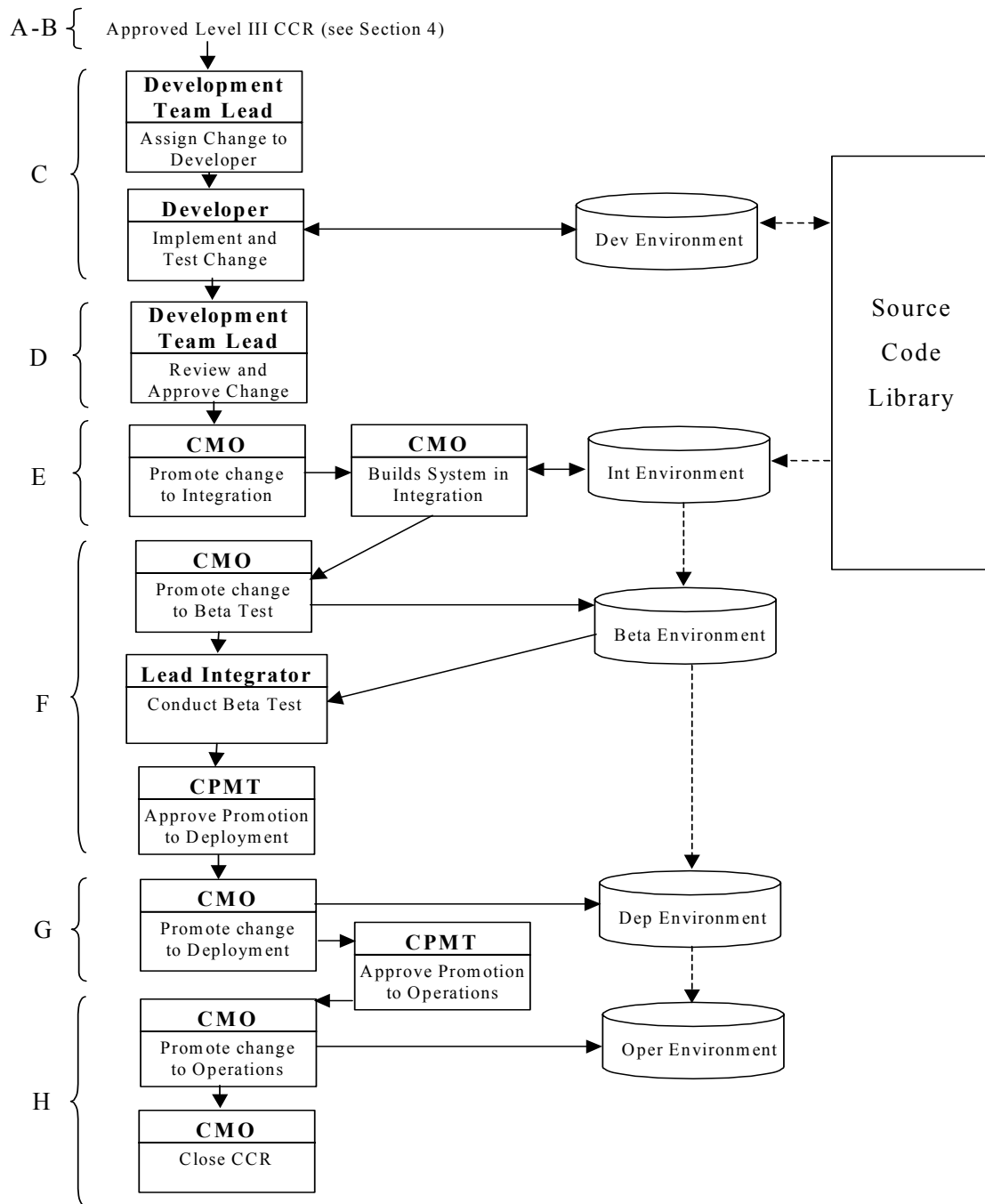


Figure 3 - Promotion Process

A- Originator Creates Configuration Change Request

A configuration change request (CCR) can be opened for any reason, for example: to report a problem or to assign a new or maintenance development task. In order to submit a CCR, the originator needs access to the Remedy Change Management tool.

When filling in a CCR, the following guidelines apply.

User id must be entered in the ID+ field. Once a valid user id is entered, the user must press enter and see his/her name populate the Name+ field. If the name field is not populated, please contact the CMO.

Enter a short description in the Short field.

Enter a detailed and complete description of the problem in the Details field.

Select CLASS-CCR for the Category Field.

Select the appropriate values for Type and Item

Select the appropriate priority.

The Lead Integrator is notified via email each time that a CCR is created. He or she reviews the CCR, verifies the priority and modifies it if necessary, and checks for duplication with existing CCRs. The Lead Integrator then classifies the Change Request as Level I, II, or III, and enters a rough estimate of the level of effort required for implementation. The Lead Integrator then assigns the CCR to the CMO for CCB action.

B- CCB Reviews Changes

The CMO prepares the list of CCRs for review at the CCB meeting, and distributes the list to the CCB members. The CCB members review each CCR and determine the disposition: accept, reject, escalate to higher level for consideration, or reassign for further analysis.

If the change is approved, the CCB assigns the CCR to a release and a development team for planning and implementation and continues with step C of the CCR process. The CMO changes the supervisor on the CCR to the development lead for the assigned development team.

If the Change is rejected, the CMO changes the status to Closed.

Emergency Changes have to be resolved within two hours of receipt. If no consensus is

reached, the Integration Facility TAL will approve or reject the change.

C- Developer Implements Change

The development lead for the assigned development team assigns the CCR to a specific developer. The developer who is assigned as a supervisor for a CCR is responsible for the CCR from the time it is assigned to him or her until it is promoted to the Integration environment. Generally, the developer assigned to supervise a particular CCR is someone who is familiar with the affected area of the system. The developer may notice that a CCR is a duplicate of an existing request, in which case the developer will relate the two changes.

The developer completes the design, code, and development testing according to the procedures defined in the CLASS Software Development Guide. After the design has been reviewed and approved, the developer checks out the modules that need to be changed and implements the changes. After the code has been reviewed and approved, the developer can commit the changes to the development repository. The developer then completes functional testing of the change in the development environment.

Each file that is modified must be noted in the CCR along with the version of that file. This is done automatically by CVS when the file is committed. The list of runtime files that will be moved into the operational environment must be added to the CCR. This information goes in the Runtime Files field of the form.

In the Instructions field of the form the developer should include:

1. Build instructions
2. Customizations required
3. Any other instruction that the developer considers necessary.

The Developer documents all changes done in the Implementation field of the Form. The Developer writes a test plan and updates the Test field with it. Expected test results should also be provided.

The Developer also updates any necessary system documentation, e.g., design documentation, test plans.

A final peer review is conducted at the completion of development testing and documentation to verify that the change is complete and documented. At the completion of this review, the developer makes the following changes to the CCR:

The State is set to Pending and the Pending reason to Supervisor Action
The Supervisor is set to Development Lead.
The Work Log is updated

D- Development Lead Reviews CCR

The Development Lead reviews the CCR, ensures that all peer reviews have been completed and that accompanying documentation is provided, and reassigns the CCR to the CMO for promotion to Integration.

E- CMO (Int) Promotes CCR

The CMO receives email stating that the CCR is ready to be promoted to Integration. The CMO:

Checks the CCR to ensure that all information is clear and correct.

Makes the necessary changes to the system Makefile. All subsystems are already included in the system Makefile, but the CMO may make changes if necessary.

If there are changes with the Environment variables, the CMO runs a script to load the new environment variables, and populates the CM database with the new values for all development and operational environments/hosts. The CMO updates the values in the database as needed and finally runs the script to unload all environment files.

Checks that all the files are under CVS control and unlocked, that the latest version is listed in the CCR, and that the version in the dev directory is the same as the one in the CVS directory. Fixes any problem found.

Runs a script to do the CVS promotion from Dev to Int.

Updates the Department field in the CCR form to Integration.

Executes the system Makefile to build the runtime files in the Int environment.

Runs a script to update the CM Inventory database.

All promotions and configuration changes performed by the scripts listed above are recorded automatically in the CM log.

F- CMO (Beta) Promotes CCR

The CMO, after verifying that the system has correctly built in the Integration environment, promotes the CCR (or group of CCRs) to the Beta Test environment

The promotion to the Beta Test environment does not involve compiling. It involves copying the runtime files to the appropriate directories.

All promotions and configuration changes performed are recorded in the CM log. The CMO changes the CCR Supervisor to the Lead Integrator.

The test team performs functional and stress testing to verify general application functionality, and regression testing to validate that the change did not adversely affect other parts of the system. Generally, tests in the Beta Test environment are performed with 'real' data.

The Lead Integrator reports results to the CPMT; the CPMT authorizes the promotion to the operational environment based on the test results. Once the change is approved for operations, the Lead Integrator changes the CCR Supervisor to the CMO.

G- CMO promotes changes to Deployment Environment

The CMO promotes the release to the deployment test environment at the NCDC operational site. This exercises the deployment process, and provides the operations personnel at the second site with the opportunity to become familiar with the new release before it becomes operational.

H- CMO puts CCR into Operation

The CMO promotes the changes into the operational environment as follows:

The CMO FTPs all the runtime files to the operational machines.

On each operational machine affected, the CMO stops the affected subsystems and servers, backs up the current runtime files, and copies the new runtime files to the runtime directories.

If there are environment file updates, the CMO updates the environment files.

Starts all the subsystems and servers.

Changes the category of the CCRs to Oper.

Monitors closely for a week to ensure that all changes are working properly.

Once the CMO is satisfied with the changes, the CMO closes the CCR.

All promotions and configuration changes performed above are recorded in the CM log.

1.4 The Remedy Change Management Tool

The Remedy Change Management Tool is the tool used by CLASS to track all change requests. To access the Change Management Tool the user needs Remedy's user tool installed in his PC.

Quick guide for Remedy CM tool

This guide assumes that you have some practice with the Remedy user tool. If you are not familiar with the tool, a little practice and some help from someone familiar with the tool should be enough. Detailed information is available in the help menu of the User tool.

Creating a Change Request

1. Login into the Remedy User Tool.
2. Open Form Remedy Application Navigator, in search mode.
3. Select Change Management Application.
4. Click Create. A new form will open.
5. Enter your user id and press enter. The Name field should populate with your name.
6. If Name field does not populate contact the CMO.
7. Enter Short Description of the Problem.
8. Enter detailed description of the Problem.
9. Select CLASS-CCR as category.
10. Select Type and Item.
11. Click on Save Button (Upper right corner of the window)

Working with a Change Request

1. Login into Remedy User Tool.
2. Open Form Remedy Application Navigator, in search mode.
3. Select Change Management Application.
4. Click Query. A new form will open.
5. Enter search data and click on Search button (Upper right corner of the window).
6. Select CCR to modify from results list.
7. Modify CCR.
8. Click on Save button (Upper right corner of the window).

I have done all the software changes, what do I do next?

1. Check-in all source code using the ci command.
2. Open CCR for Change following the instructions above.

3. In the Implementation field enter a full detail of all changes made.
4. In the Test field enter the Test plan and expected results.
5. In the Instructions field enter build instructions and other special instructions that the CMO or Test Manager should be aware of.
6. Verify that all Source code changes and revision numbers are in the Source Code field.
7. Enter all runtime files in the Runtime Files field.
8. Change Group to SAA-SW-Manager
9. Change Supervisor to Developer Manager
10. Change Status to Pending
11. Change Pending reason to Supervisor Action
12. Update Work Log.
13. Save Your Changes.